

(12) UK Patent Application (19) GB (11) 2 307 571 (13) A

(43) Date of A Publication 28.05.1997

(21) Application No	9623606.2	(51) INT CL ⁶	G06F 17/27 17/30
(22) Date of Filing	13.11.1996	(52) UK CL (Edition O)	G4A AUDB
(30) Priority Data		(56) Documents Cited	
(31) 07306205	(32) 24.11.1995	(33) JP	EP 0620527 A2 Dialog record 01805820 of Windows Sources, v3, n8, p128(3), August 1995
(71) Applicant(s)	Dainippon Screen Mfg Co Ltd (Incorporated in Japan)	(58) Field of Search	UK CL (Edition O) G4A AUDB INT CL ⁶ G06F 17/21 17/27 17/30 Online databases: COMPUTER, INSPEC, WPI
(72) Inventor(s)	Toru Takasawa Hiroshi Nakayama Hideaki Kitamura		
(74) Agent and/or Address for Service	David Keltie Associates 12 New Fetter Lane, LONDON, EC4A 1AP, United Kingdom		

(54) Automatically generating a document type definition

(57) A document type definition (hereafter referred to as DTD) of SGML (Standard Generalised Markup Language) is automatically generated. A user prepares a plurality of sample documents provided with tags in accordance with the syntax of SGML about a document type for which the user wants to create a DTD. These sample documents are input from a sample inputting unit 10 and a structure information extracting unit 12 extracts logical structure information from the tags of the sample documents. A structure list processing unit 14 creates a structure list in accordance with the extracted information. When processing of all prepared sample documents is completed, a DTD creating unit 16 generates a DTD by analyzing the structure list.

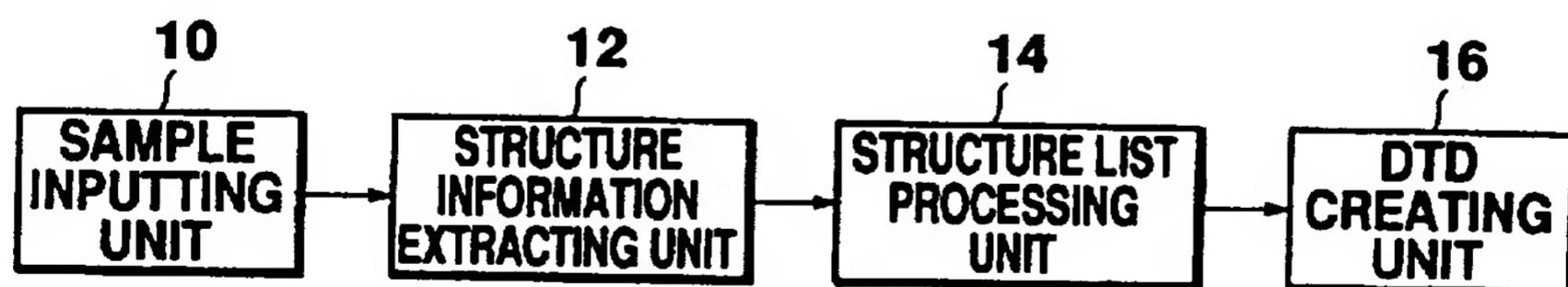


Fig. 1

GB 2 307 571 A

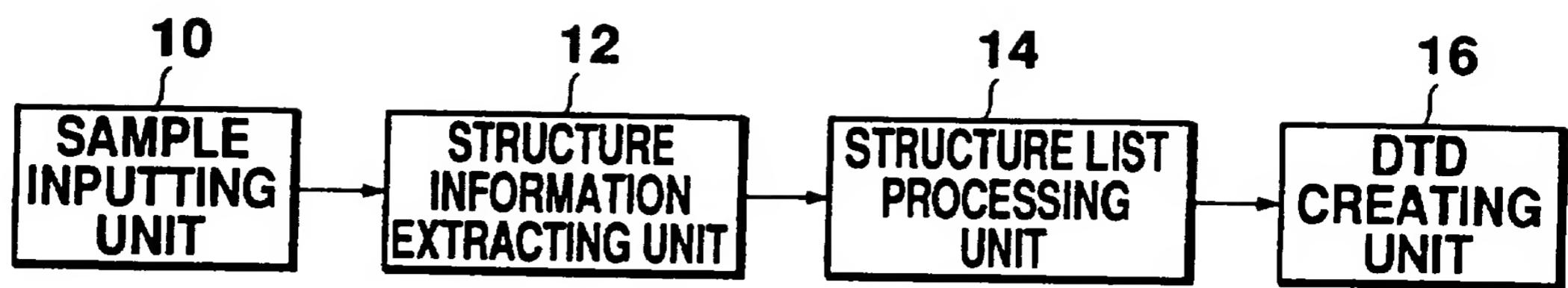


Fig. 1

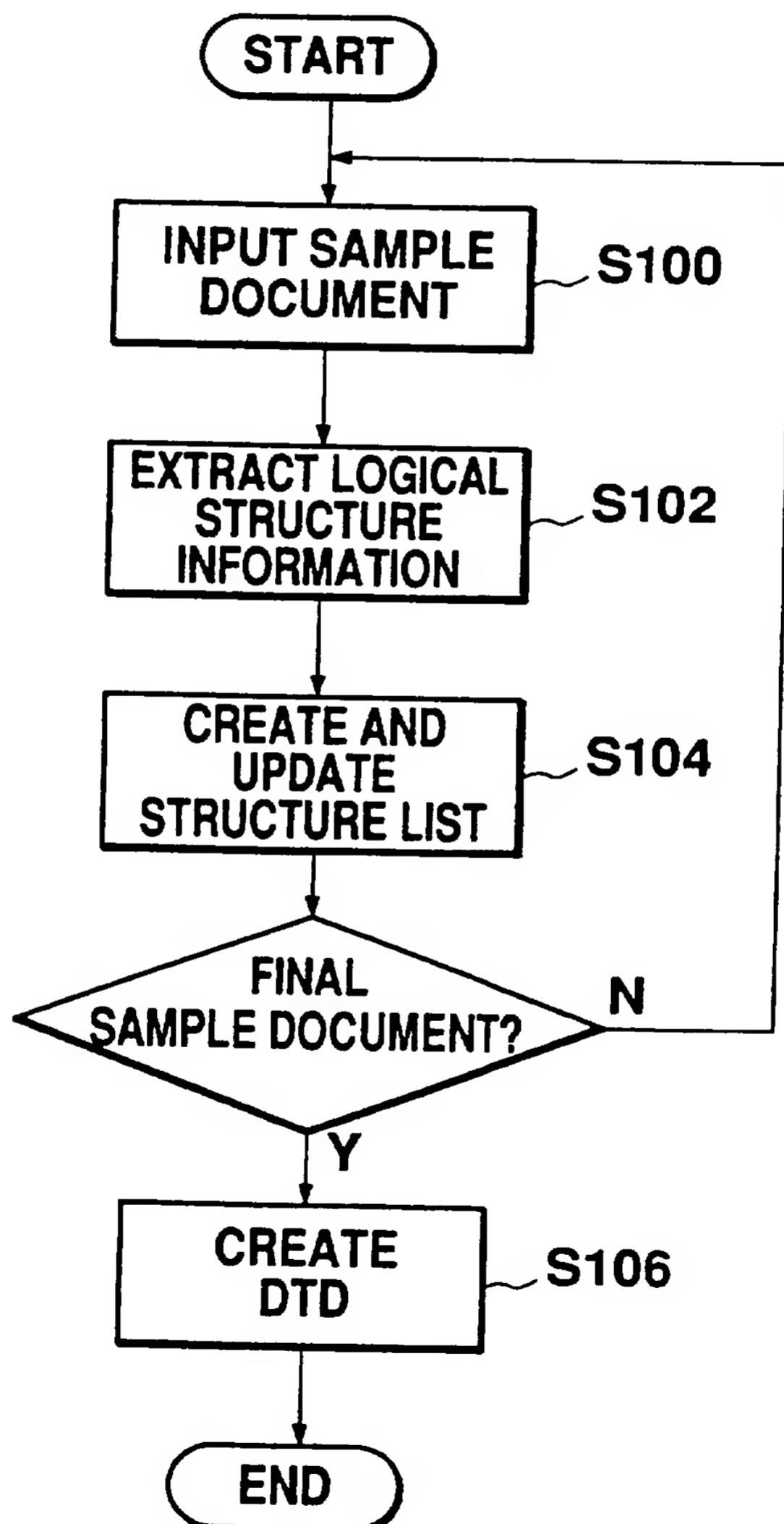


Fig. 2

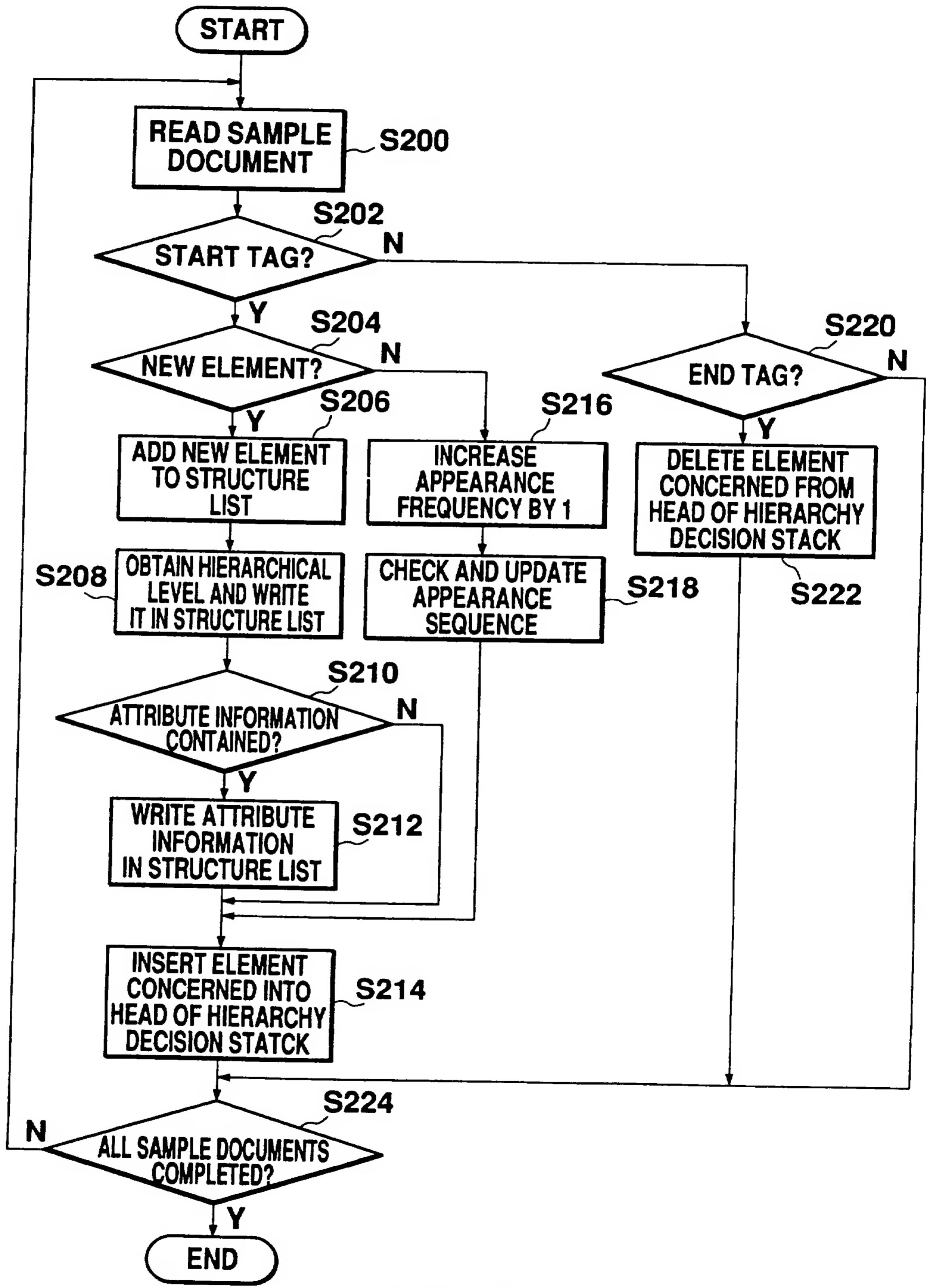


Fig. 3

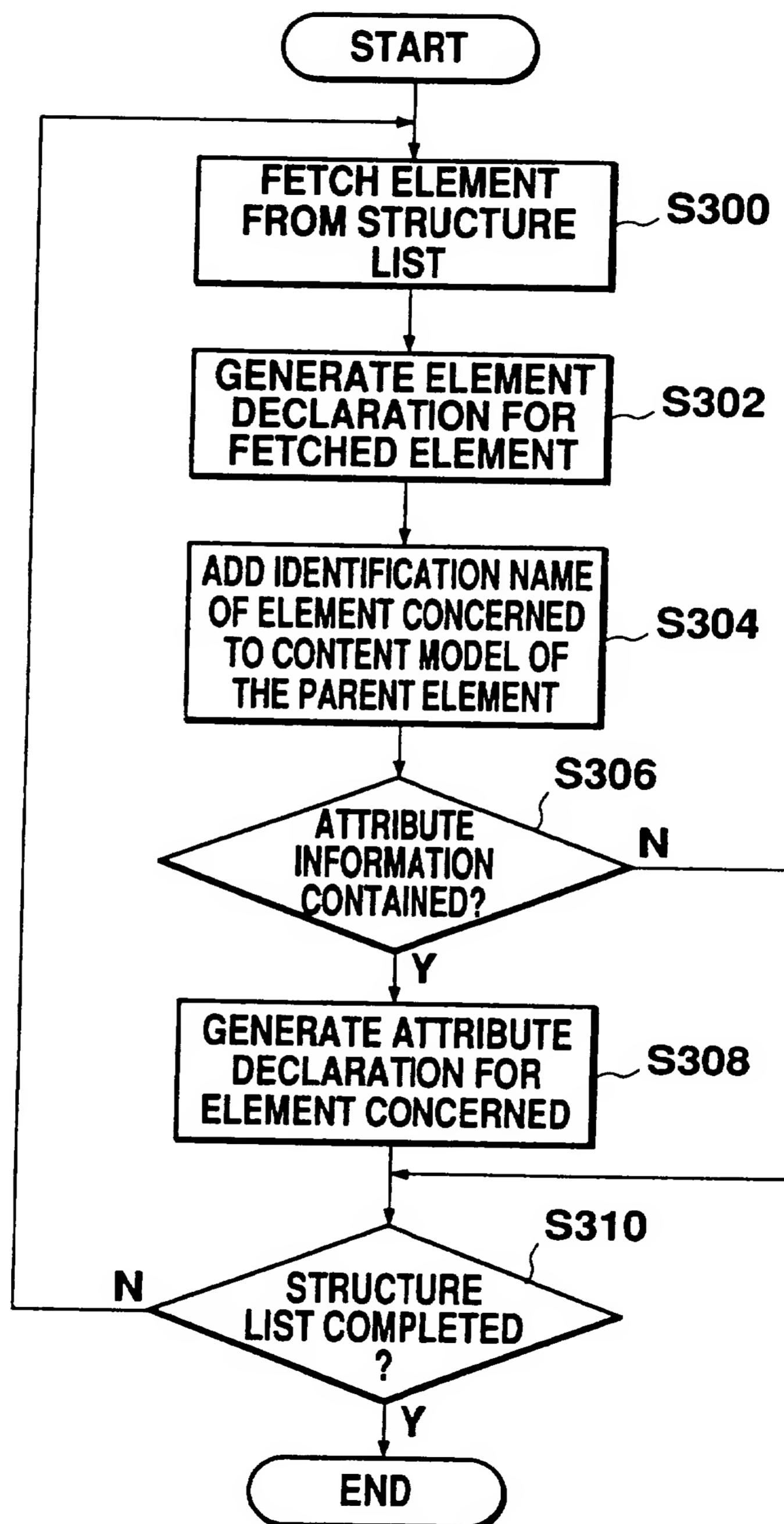


Fig. 4

Fig. 5

[Sample document 1]

```

<Concert-infor.>
<Artist-name genre = Jazz> Ohtsua Taro </Artist-name>
<Opening-date> 5/27, 6/2, 6/23 </Opening-date>
<Place>
<Venue-name> Ikebukuro Hall </Venue-name>
<Address> 2-3, Kami-ikebukuro 1-chome, Toshima-ku </Address>
<Telephone> 123-4567 </Telephone>
</Place>
</Concert-infor.>

```

Fig. 6

[Sample document 2]

```

<Concert-infor.>
<Artist-name genre = Rock> Ikebukuro Hanako </Artist-name>
<Opening-date> 5/23 to 6/4 </Opening-date>
<Place>
<Venue-name> Tokyo Stadium </Venue-name>
<Address> 2-3, Yamabukuro 1-chome, Toshima-ku </Address>
<Telephone> 234-5678 </Telephone>
</Place>
</Concert-infor.>

```

Fig. 7

[Sample document 3]

```

<Concert-infor.>
<Artist-name genre = Classic> Jiro Tokyo </Artist-name>
<Place>
<Venue-name> Toshima Hall </Venue-name>
<Address> 2-3, Umibukuro 1-chome, Toshima-ku </Address>
<Telephone> 345-6789 </Telephone>
</Place>
<Opening-date> 6/12 to 6/18 </Opening-date>
<Cost> ¥10,000 </Cost>
</Concert-infor.>

```

Fig. 8

Identifier	Identification name	Link information	Appearance frequency	Appearance sequence	Attribute information	Hierarchical level
0	Concert-infor.	S, 1	1	0		0
1	Artist-name	0, 2	1	0		1
2	Opening-date	1, 3	1	0		1
3	Place	2, 4	1	0		1
4	Venue-name	3, 5	1	0		1
5	Address	4, 6	1	0		2
6	Telephone	5, e	1	0		2

Fig. 9

Identifier	Identification name	Link information	Appearance frequency	Appearance sequence	Attribute information	Hierarchical level
0	Concert-infor.	S, 1	2	0		0
1	Artist-name	0, 2	2	0		1
2	Opening-date	1, 3	2	0		1
3	Place	2, 4	2	0		1
4	Venue-name	3, 5	2	0		2
5	Address	4, 6	2	0		2
6	Telephone	5, e	2	0		2

Fig. 10

Identifier	Identification name	Link information	Appearance frequency	Appearance sequence	Attribute information	Hierarchical level
0	Concert-infor.	S, 1	3	0		0
1	Artist-name	0, 2	3	0		1
2	Opening-date	1, 3	3	0		1
3	Place	2, 4	3	0		1
4	Venue-name	3, 5	3	0		1
5	Address	4, 6	3	0		2
6	Telephone	5, 7	3	0		2
7	Cost	6, e	1	0		2

Fig. 11

```
<! ELEMENT Concert-infor. - - (Artist-name, (Opening-date & place), Cost?)>
<! ELEMENT Artist-name - - (#PCDATA)>
<! ATTLIST Artist-name genre id ID #IMPLIED >
<! ELEMENT Opening-date - - (#PCDATA)>
<! ELEMENT Place - - (Venue-name, Address, Telephone)>
<! ELEMENT Venue-name - - (#PCDATA)>
<! ELEMENT Address - - (#PCDATA)>
<! ELEMENT Telephone - - (#PCDATA)>
<! ELEMENT Cost - - (#PCDATA)>
```

Fig. 12

```

<Receipt>
<Header> Receipt </Header>
<Content>
<Date> October 15, 1995 </Date>
<Money-destinat.> Store A </Money-destinat.>
<Amount-of-money> ¥20,600 </Amount-of-money>
<Comment-1> As the price of goods x </Comment-1>
<Comment-2> The above amount is correctly received. </Comment-2>
</Content>
</Receipt>

```

Fig. 13

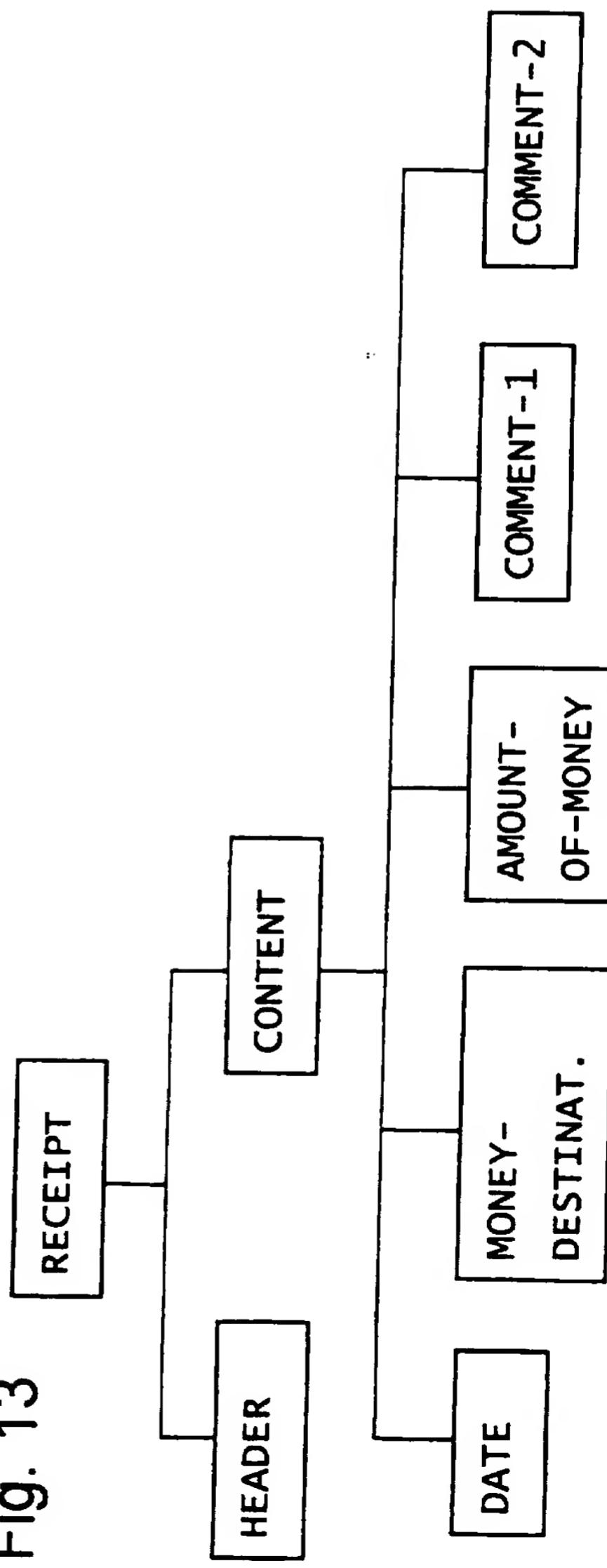


Fig. 14

```
<! ELEMENT Receipt
  <! ELEMENT Header
  <! ELEMENT Content
  <! ELEMENT Date
  <! ELEMENT Money-destinat.
  <! ELEMENT Amount-of-money
  <! ELEMENT Comment-1
  <! ELEMENT Comment-2>
  - - (Header, Content)>
  - - (#PCDATA)>
  - - (Date, Money-destinat., Amount-of-money, Comment-1, Comment-2)>
  - - (#PCDATA)>
  - - (#PCDATA)>
```

APPARATUS AND METHOD FOR GENERATING DOCUMENT TYPE DEFINITION

Background of the Invention

[Field of the Invention]

The present invention relates to a document type definition generating apparatus for generating a document type definition used for controlling a structured document such as an SGML document.

[Related Arts]

SGML (Standard Generalized Markup Language) is an international standard for a document description language created to simplify the exchange of similar documents through an electronic medium or their storage a databases for reuse.

In the case of SGML, a document is divided into several elements to describe the logical structure of the document as a hierarchical relation between the elements. Each element in the document is provided with a tag including information such as the identification name or attribute of the element and the document processing system of SGML processes the document using the tag as control information.

For example, Fig. 12 shows a receipt described with SGML. In Fig. 12, the receipt is divided into such elements as "header", "content", and "date". "Header" or "content" serves as the identification name of an element, and the start and end of elements are shown by respective start

"<element identification name>" and end tags "</element identification name>". Moreover, the hierarchical relation of an element is shown as a nesting structure of the element. That is, when an element is described in another element, the former and the latter have the relation of a child and a parent. For example, because the element "date" is described between the start tag <content> and the end tag </content> of the element "content", a hierarchical relation is formed that "content" is a parent element of "date". Therefore, Fig. 12 shows the logical structure shown in Fig. 13. For SGML, various additional rules, including those for describe the attribute of elements, are specified. However, their description is omitted in this specification. Each document provided with tags in accordance with the syntax of SGML as described above is referred to as a document instance.

The document processing system of the SGML controls each document instance by using a document type definition (hereafter referred to as DTD). The DTD defines rules for providing tags when describing a document in accordance with the SGML. The DTD is created for each type of document. For example, the DTD of a receipt is shown by the format in Fig. 14. The DTD comprises such declarative statements as an element declaration ("<!ELEMENT...>" in Fig. 14) specifying the content of each element such as the

identification name of the element or child element of the element, and an attribute declaration ("<!ATTLIST...>" not illustrated in Fig. 14) specifying the attribute of an element. Moreover, an attribute is a piece of information only used for computer processing and does not appear on a printout or display.

The document processing system of the SGML checks the syntax of a document instance created by a user, supports the creation of a document instance according to an interactive technique, and retrieves data. In the case of a document of a type where the DTD is entered in the document processing system, it is possible to easily create or retrieve a document instance by using the processing function.

Therefore, to process a document in accordance with SGML, it is necessary to previously create the DTD for a document type to be processed. Therefore, the logical structure common to documents included in the document type has previously been analyzed and the analysis results have been coded to the format shown in Fig. 14. The above coding has until now been manually performed by a SGML specialist because specialized knowledge is required for SGML.

Therefore, it is difficult for an ordinary user to create and process a document of a new document type in accordance with the SGML, to easily convert the new document

type to SGML at the user level because analysis of the logical structure and coding by a specialist are necessary. Thus creation of the DTD is a bottleneck of progressing the use of SGML, and a tool for efficiently creating the DTD is desired.

Summary of the Invention

The present invention is made to solve the above problems and its object is to provide a document type definition generating apparatus for automatically generating a document type definition (DTD) for a structure description document such as SGML.

To achieve the above object, a document type definition generating apparatus of the present invention comprises structure information extracting means for extracting logical structure information from a sample document described in accordance with a structure description language and definition creating means for creating a document type definition in accordance with the extracted logical structure information.

In the case of the present invention, structure information extracting means extract logical structure information from a sample document created in accordance with the syntax of a structure description language on a document type for which a document type definition will be created and definition creating means generate a document

type definition with a predetermined format from the extracted logical structure information. The present invention makes it possible for a user to obtain a document type definition simply by preparing a sample document showing the image of a desired document; detailed knowledge of the structure of the document type definition is not necessary. It is also possible for a user to create a sample document or use a sample document selected out of existing documents.

Moreover, the document type definition generating apparatus of claim 1 of the present invention includes totalizing means for totalizing logical structure information extracted from a plurality of sample documents by the structure information extracting means, in which the definition creating means generates a document type definition in accordance with the results totalized by the totalizing means.

In the case of the above structure, logical structure information is extracted from each of a plurality of sample documents created for the same document type and a document type definition is created in accordance with the result obtained by totalizing the logical structure information. The above structure makes it possible to automatically generate a document type definition which can be commonly used for a plurality of sample documents. Therefore, when a

user requires the document type definition of a certain document type, he/she can obtain that document type definition of the document type by describing several actual examples in accordance with the syntax of a structure description language as sample documents or by preparing document examples already created for the document type as sample documents and inputting the sample documents to the apparatus of the present invention.

Moreover, a document type definition generating apparatus of the present invention is characterized in that the definition creating means creates a document type definition in accordance with the totalization result obtained whenever logical structure information from each sample document is totalized.

In the case of the above structure, an updated document type definition is created whenever a sample document is input. Therefore, a user can efficiently obtain a document type definition intended by the user.

A document type definition generating apparatus of the present invention can use the SGML as a structure description language. In this case, structure information extracting means recognize the tag of a sample document and extracts logical structure information from the document.

Moreover, in the case of a document type definition creating apparatus of the present invention, the structure

information extracting means analyze the nesting structure of each element of a sample document to obtain the hierarchical relation between elements.

The structure information extracting means extracts the information for the hierarchical relation between elements from the logical structure information of a document by analyzing the nesting structure of the elements. Thus, the apparatus can automatically extract the hierarchical relation between the elements of a document.

When SGML is used as a structure description language, structure information extracting means extracts the hierarchical relation between elements from start and end tags by analyzing the nesting structure of each element.

Brief Description of the Drawings

Figure 1 is a functional block diagram of a document type definition generating apparatus of the present invention;

Figure 2 is a flowchart showing the general process of a document type definition generating apparatus of the present invention;

Figure 3 is a flowchart showing the process of creating and updating a structure list;

Figure 4 is a flowchart showing the process of automatically generating a DTD from a structure list;

Figure 5 is an illustration showing sample document 1;

Figure 6 is an illustration showing sample document 2;
Figure 7 is an illustration showing sample document 3;
Figure 8 is an illustration showing a structure list
created from sample document 1;

Figure 9 is an illustration showing a structure list
created from sample documents 1 and 2;

Figure 10 is an illustration showing a structure list
created from sample documents 1, 2, and 3;

Figure 11 is an illustration showing a DTD created in
accordance with the structure list in Fig. 10;

Figure 12 is an illustration showing a receipt
described in accordance with SGML;

Figure 13 is an illustration showing the structure of
the document instance in Fig. 12; and

Figure 14 is an illustration showing a DTD of a
receipt.

Description of the Preferred Embodiments

Embodiments of a document type definition generating
apparatus of the present invention are described by
referring to the accompanying drawings.

A document type definition generating apparatus of the
present invention receives a sample document prepared by a
user as an input and extracts logical structure information
from the sample document to automatically generates a
document type definition (DTD). A document provided with a

tag in accordance with the syntax of SGML is used as the sample document. It is possible for a user to create a sample document or use a sample document selected out of existing document instances.

Figure 1 is a functional block diagram of a document type definition generating apparatus of the present invention and Figure 2 is a flowchart showing operations of the document type definition generating apparatus in Fig. 1. This embodiment is described below by referring to Figs. 1 and 2.

Sample documents prepared by a user are input from sample inputting unit 10 (S100). Structure information extracting unit 12 extracts logical structure information from the tags of the input sample documents (S102). The logical structure information extracted from the sample documents includes identification names of elements in the sample documents and information for the hierarchical relation between the elements. Structure list processing unit 14 creates or updates a structure list in accordance with the extracted logical structure information (S104). The structure list is a list used for this embodiment to totalize the logical structure information extracted from the sample documents. Specific examples of the structure list and detailed operations of structure information extracting unit 12 and structure list processing unit 14 are

described later in detail.

The cycle of input of the above sample documents, extraction of logical structure information, and update of the structure list is repeated as long as sample documents prepared by the user remain. After all the prepared sample documents are processed, structure list processing unit 14 outputs a final structure list to DTD creating unit 16. DTD creating unit 16 generates a DTD from the final structure list (S106). Detailed operations of DTD creating unit 16 are described later in detail.

The apparatus of this embodiment comprises the above structure which makes it possible to automatically generate a DTD for describing the structure of the prepared sample documents.

The process of creating and updating a structure list and generating a DTD from the structure list in this embodiment is described below using specific examples. A document of "concert-infor." is describe below as an example.

When a user wants to create and control SGML "concert-infor." documents, he creates some actual examples as sample documents. The sample documents are described by adding start and end tags in accordance with the syntax of SGML. A plurality of sample documents thus created are input to sample inputting unit 10. A case is described below in

which three document instance shown in Figs. 5 to 7 are input in order as sample documents.

Structure information extracting unit 12 reads the tag of a sample document supplied from sample inputting unit 10 and extracts logical structure information from the sample document. Then, structure list processing unit 14 adds the extracted logical structure information to a structure list.

The structure list of this embodiment, as shown in Fig. 8, has seven fields of element identifier, identification name, link information, appearance frequency, appearance sequence, attribute information, and hierarchical level. In this case, the identifier denotes an ID number added to an element of a document. The link information is used to show the link relation between elements in the structure list, showing the identifier of an element linked immediately before the element concerned and the identifier of an element linked immediately after the element concerned. Symbols "s" and "e" in the link information respectively denote the head and the tail of a list. The appearance frequency denotes the number of times for the element concerned to appear in a plurality of input sample documents. For example, because Fig. 8 shows a structure list created when the first sample document is read, the appearance frequency of each element is 1. The appearance sequence field shows whether the appearance sequence of the

element concerned has been replaced with that of other element in a plurality of input sample documents. Attribute information of the element concerned is written in the attribute information field. The attribute information is written next to the identification name in the start tag and used as a key for retrieval or the like. For example, "genre" written in the start tag of the element "artist-name" of the sample document in Fig. 5 is attribute information. The hierarchical level is the value showing the hierarchical relation between the element concerned and other element. In the case of the structure list in Fig. 8, for example, the element "concert-infor." at hierarchical level 0 is the parent of the subsequent elements "artist-name", "opening-date", and "place" at hierarchical level 1, and the elements "venue-name", "address", and "telephone" at hierarchical level 2 are children of the element "place" at hierarchical level 1 immediately before those elements.

Figure 8 shows the state of the structure list when processing of sample document 1 (Fig. 5) is completed and Figs. 9 and 10 show the states of structure lists when processing of sample document 2 (Fig. 6) and that of sample document 3 (Fig. 7) are completed respectively.

The process of creating and updating a structure list by structure information extracting unit 12 and structure list processing unit 14 is described below in detail by

referring to the flowchart in Fig. 3.

Structure information extracting unit 12 sequentially reads character strings of a sample document (S200). Then, unit 12 decides whether a read character string is a start tag (S202). That is, when the read character string starts with a tag starting delimiter "<" and an identification name comes next to the delimiter "<" with no slash "/", unit 12 decides that a character string until a tag ending delimiter ">" comes is a start tag.

When the read character string is a start tag, unit 12 reads an element identification name entered at the head of the start tag, check whether the element identification name is already entered in the structure list, and decides whether the element concerned is a new element (S204). That is, structure information extracting unit 12 refers to the current structure list present in structure list processing unit 14, and decides that the element concerned is a new element when the element identification name concerned is not entered in the structure list and that the element concerned is not a new element when it is already entered in the list. The decision results are transferred to structure list processing unit 14.

When it is decided that the element concerned is a new element, structure list processing unit 14 adds the new element to the structure list (S206). That is, unit 14

gives an element identifier to the new element concerned, writes the element identifier and element identification name in the structure list, and sets the appearance frequency of the element concerned to 1. Moreover, structure list processing unit 14 updates link information in accordance with the above addition.

Then, structure list processing unit 14 obtains the hierarchical level of the element concerned to write it in the structure list (S208). The hierarchical level is obtained by using a data structure of LIFO (Last In First Out) referred to as a hierarchy decision stack. The procedure for obtaining a hierarchical level using the hierarchy decision stack is described below.

In the case of this embodiment, data is inserted into the head of the hierarchy decision stack when reading a start tag and the data at the head of the hierarchy decision stack is deleted when reading an end tag (S222). The identification name of an element and the hierarchical level of the element are inserted into the hierarchy decision stack as stack data of the element. Because of the above structure, the head of the hierarchy decision stack shows the latest element among uncompleted elements (that is, elements not closed by an end tag). Therefore, a newly read element is a child of the element at the head of the hierarchy decision stack. When a new element is added to

the structure list, unit 14 reads the hierarchical level of the element at the head of the hierarchy decision stack in S208 and sets a value obtained by adding 1 to the hierarchical level as the hierarchical level of the new element.

Specifically, because the hierarchy decision stack is empty when the first element "concert-infor." of sample document 1 in Fig. 5 is input, unit 14 sets the hierarchical level of "concert-infor." to 0 as shown by the structure list in Fig. 8. Then, in S214, unit 14 inserts "concert-infor." and its hierarchical level 0 to the head of the hierarchy decision stack and still continuously reads the sample document. When the start tag of "artist-name" comes, the data of "concert-infor." and the data of hierarchical level 0 are present at the head of the hierarchy decision stack because the end tag of "concert-infor." does not yet come. Therefore, the hierarchical level of "artist-name" is set to 1. Thereafter, in S214, "artist-name" is also inserted into the hierarchy decision stack. In Fig. 5, because the end tag of "artist-name" comes before the start tag of the next element comes, the data for "artist-name" is deleted from the hierarchy decision stack. As a result, "concert-infor." is set to the head of the hierarchy decision stack when the hierarchical level of the next element "opening-date" is decided. Therefore, the

hierarchical level of "opening-date" is also set to 1.

Hereafter, a hierarchical level is obtained from the data at the head of the hierarchy decision stack in the same manner as the above.

When writing of hierarchical levels is completed, unit 14 detects whether attribute information is included in read start tags (S210). When attribute information is included, unit 14 writes the information in the attribute information field of the structure list (S212). For example, unit 14 writes "genre" in the attribute information field of "artist-name" in step S212 because attribute information "genre" is described in the start tag of the element "artist-name" of the sample document in Fig. 5. Thereafter, as described above, unit 14 inserts the identification name of the element concerned and the hierarchical level of the element concerned obtained in S208 into the head of the hierarchy decision stack (S214).

The flow of the processing (S206 to S214) when it is decided that the read element is a new element in accordance with the decision in S204 is described above. However, when it is decided in S204 that the read element is not a new element, unit 14 adds 1 to the appearance frequency of the element concerned in the structure list (S216). Unit 14 checks then whether the appearance turn of the element concerned is different from the appearance turn in the last

document. When the appearance turn of the element is different from the appearance turn in the last document, the unit 14 writes a value showing that the element concerned is replaced with other elements in sequence in the appearance sequence field. For example, because "opening-date" is replaced with "place" in sequence in the case of sample document 3 (Fig. 7), the value 1 showing that sequences are replaced is written in the field of appearance sequence of "opening-date" and "place" in the case of the structure list (Fig. 10) after processing of sample document 3 is completed. Even when the read element is not a new element, the data for the element concerned is inserted into the hierarchy decision stack in order to obtain hierarchical information (S214).

A case is described below in which it is decided in S202 that a character string read out of a sample document is not a start tag. In this case, it is first decided whether the character string is an end tag (S220). When the read character string starts with a tag start delimiter "<" and a slash "/" comes next to the delimiter, it is decided that a character string until an end delimiter ">" comes next is an end tag. When the character string is decided as an end tag, the data for the element concerned is deleted from the head of the hierarchy decision stack (S222). However, when it is decided in S220 that the character

string is not an end tag, no operation is performed because the character string shows document content other than a tag.

When the above processing is completed, it is decided whether all sample document processing is completed (S224). If processing is not completed, the next character string is read out of a sample document and the above process is repeated. When processing of every sample document is completed, creation and update of structure lists are completed, and a final structure list is input to DTD creating unit 16 from structure list processing unit 14.

States of the structure lists shown in Figs. 8 to 10 are briefly described below. When processing of sample document 1 (Fig. 5) is completed, the appearance frequency of element shown in Fig. 8 is set to 1 in the structure list. In the case of sample document 2 (Fig. 6) to be next input, the same types of elements as the case of sample document 1 appear in the same sequence. Therefore, when processing of sample document 2 is completed, the appearance frequency of every element in the structure list is set to 2 as shown in Fig. 9. Though every element present in sample documents 1 and 2 appears in sample document 3 (Fig. 7), the appearance turns of "opening-date" and "place" are replaced with those of sample documents 1 and 2. Moreover, the element "cost" is added to sample document 3, which is not

present in sample documents 1 and 2. Therefore, when processing of sample document 3 is completed, the appearance frequency of newly added "cost" is set to 1 and the appearance frequencies of elements other than "cost" are set to 3. Because the element "cost" is a child element of "concert-infor.", the hierarchical level of "cost" is set to 1. The value 1 showing replacement was made is written in the appearance sequence field of "opening-date" and "place".

Creation and update of structure lists by structure information extracting unit 12 and structure list processing unit 14 are described above. DTD creating unit 16 analyzes the final structure list obtained as the result of the creation and update process and creates a DTD for comprehensively specifying logical structures of sample documents. The process of DTD creation in DTD creating unit 16 is described below by taking a case of creating a DTD from the structure list shown in Fig. 10 as an example.

Figure 4 is a flowchart showing the process in DTD creating unit 16. DTD creation process is described below by referring to the flowchart. The DTD finally created from the structure list in Fig. 10 has the form shown in Fig. 11. The following description is continued while properly referring to Fig. 11.

As shown in Fig. 4, DTD creating unit 16 fetches an element from a structure list and generates an element

declaration for the fetched element (S302). When the element has a parent, unit 16 adds the identification name of the element to the content model of the parent element (S304). In this case, the content model is a list showing identification names of child elements in order from the left in accordance with their appearance sequence. For example, in Fig. 11, the portions written in parentheses at the latter half of the element declaration of "concert-infor." at the first line of the DTD serves as the content model of the element "concert-infor.". Then, unit 16 checks whether attribute information is included in the element concerned (S306). When the attribute information is included, unit 16 generates an attribute declaration for the element concerned (S308). For example, in Fig. 11, the third line of the DTD serves as an attribute declaration for the element "artist-name". After the above series of processing, unit 16 decides whether processing up to the final structure list is completed (S310). When the processing is not completed, unit 16 fetches the next element of the structure list to repeat the above processing. By processing up to the final structure list, a DTD reflecting the contents of structure lists can be obtained.

In the case of the structure list in Fig. 10, data such as identification name, appearance frequency, appearance

sequence, attribute, and hierarchical level is fetched from the first element "concert-infor." (S300) and an element declaration of the element "concert-infor." is generated by using the identification name. As a result, the element declaration <!ELEMENT concert-infor.> is created as the first line of the DTD. However, the content model at the latter half of the element declaration of "concert-infor." shown in Fig. 11 is not created at this stage. Because its hierarchical level is 0, "concert-infor." does not have a parent element that is, it is the most significant element and therefore, step S304 processing is not performed. Moreover, because no attribute information is described about "concert-infor." in the structure list, S308 processing is also not performed.

Thus, when processing of the first element "concert-infor." of the structure list is completed, unit 16 fetches the data for the next element "artist-name" from the structure list (S300). Unit 16 then creates an element declaration for "artist-name" as the second line of the DTD (S302). Unit 16 writes the element "artist-name" in the content model of the element declaration of the parent element (S304). That is, because the hierarchical level of the element "artist-name" is set to 1, unit 16 decides that the element (in this case, "concert-infor.") with hierarchical level 0 appearing at the closest position

before the element concerned in the structure list is a parent element and subsequently creates an area for a content model in the element declaration of the parent element "concert-infor." and writes the identification name "artist-name" in that area. Hereafter, as processing of other elements of the structure list are performed, identification names of child elements are written in the content model of the parent element in accordance with the appearance order starting with the left. In the content model, child elements are delimited by a comma ",".

Moreover, because attribute information "genre" is attached to the element "artist-name", unit 16 creates the attribute declaration of the element "artist-name" as the third line of the DTD (S308). In Fig. 11, "id ID" and "#IMPLIED" shown at the latter half of the attribute declaration of "artist-name" are an attribute type and a default value for specifying the composition of the attribute information "genre", each of which is one of the items specified in the rules of SGML. In the case of this embodiment, the attribute type and the default value are written in an attribute declaration.

When processing of the element "artist-name" is completed in accordance with the above procedure, unit 16 fetches data for the next element "opening-date" from the structure list. Thereafter, unit 16 repeats the above steps

until processing of the final element "cost" in the structure list is completed.

In the case of the structure list in Fig. 10, the appearance sequences of elements "opening-date" and "place" are set to the value "1" showing that sequences are replaced. Therefore, it is necessary to reflect the information for sequence on the DTD. In the case of SGML, replacement of sequences of elements is shown by symbol "&" in the content model of the parent element of those elements. That is, for this example, it is possible to show that elements "opening-date" and "place" can be replaced each other in sequence by writing "(opening-date & place)" in the content model of the element "concert-infor." as shown in Fig. 11.

Therefore, the unit 16 confirms the value of the appearance sequence of an element when writing the identification name of the element in the content model of the parent element of the element in S304. Because it is confirmed in the processing of the element "opening-date" that the value of the appearance sequence is set to "1", this embodiment writes "(opening-date &" in the content model of the parent element "concert-infor.". Then, this embodiment confirms that the appearance sequence of the element "place" is set to 1 when processing the element "place" appearing next to the same hierarchical level as the

element "opening-date" and writes "place)" in the content model of the parent element "concert-infor.". Thus, the information for replacement of appearance sequences is reflected on the DTD.

In the case of the DTD shown in Fig. 11, symbol "?" is added after the final child element "cost" in the content model of the element declaration of "concert-infor.". This symbol "?" shows that appearance is optional (that is, appearance is unnecessary). For this example, the appearance frequency of the element "cost" is set to 1 in the structure list in Fig. 10 and moreover, the frequency is lower than the appearance frequency of the element "concert-infor." of the most significant hierarchy. Therefore, it is decided that the appearance of the element "cost" is optional. This decision is performed in S304. That is, when this embodiment writes the identification name of an element in the content model of the parent element in S304, if the appearance frequency of the element is lower than that of the element of the most significant hierarchy, the unit 16 adds the symbol "?" after the identification name of the element and writes it in the content model of the parent element.

Moreover, a description where "(#PCDATA)" is present in the element declaration of the DTD in Fig. 11, which shows that the data type of a text written in the element is

character data to be parsed. For this data type, rules are specified in SGML. In this embodiment, #PCDATA is declared as default data type for an element including a text.

In the above description, a DTD is automatically generated from a plurality of sample documents. However, the apparatus of this embodiment can automatically generate a DTD from only one sample document.

In the above description, a DTD is generated only when every sample document is input and structure list updating is completed. However, it is also possible to generate a DTD whenever a sample is input. In this case, a user can efficiently obtain intended DTDS.

As described above, this embodiment allows even a user who does not know how to create a DTD to automatically generate a DTD from a sample document created in accordance with the syntax of SGML by preparing the sample document.

Moreover, this embodiment makes it possible to create an individual document instance before creating a DTD and create the DTD by using the created document instance as a sample document. Therefore, it is possible to simultaneously create document instances and a DTD.

Each unit shown in Fig. 1 can be realized by executing a program in which each function is described by a computer's CPU and memory. For example, sample inputting unit 10 reads a file of sample documents out of a storage

medium such as a floppy disk or hard disk and loads the file in a work area of a main memory. Logical structure extracting unit 12 extracts logical structure information by analyzing the sample document file on the main memory and structure list creating unit 14 stores the extracted result in the storage area of a structure list secured on the main memory. Then, DTD creating unit 16 reads the structure list out of the main memory to analyze it and creates a DTD in accordance with the analysis result.

However, the present invention is not restricted to the above units which are realized by a software-type method. It is also possible to realize each unit of this embodiment by a hardware circuit.

What is claimed is:

1. An apparatus for generating a document type definition of a structured document described in accordance with a structure description language, the apparatus comprising:

structure information extracting means for extracting logical structure information from a sample document described in accordance with a structure description language; and

definition creating means for creating a document type definition in accordance with the logical structure information.

2. The apparatus according to claim 1, further comprising totalizing means for totalizing logical structure information extracted by said structure information extracting means from a plurality of sample documents,

wherein said definition creating means generates a document type definition in accordance with a totalization result of said totalizing means.

3. The apparatus according to claim 2, wherein said totalizing means adds said extracted logical structure information to totalization results already stored whenever logical structure information is extracted from a sample document and said definition creating means generates a document type definition in accordance with the

totalization results obtained by said totalizing means.

4. The apparatus according to claim 1, wherein
said structure extracting means obtains the
hierarchical relation between said elements by dividing a
sample document into a plurality of elements and analyzing
the nesting structure of each of said elements in the
description of said sample document and outputs the
information for said hierarchical relation as one piece of
logical structure information.

5. The apparatus according to claim 1, wherein
said structure description language is SGML, and
said structure information extracting means extracts
logical structure information from tags included in a sample
document described in accordance with SGML.

6. The apparatus according to claim 5, wherein
said structure information extracting means dissolves a
sample document into a plurality of elements in accordance
with detection of the start tag and/or end tag in said
sample document, analyzes the nesting structure of each of
said elements in accordance with the sequence relation of
the start tag and/or end tag of each of said elements in the
description of said sample document, obtains the logical
hierarchical relation between said elements in accordance
with the analysis result, and outputs the information for
the hierarchical relation as one piece of logical structure

information.

7. A method of generating a document type definition of a structured document described in accordance with a structure description language, the method comprising:

the structure information extracting step of extracting logical structure information from a sample document described in accordance with a structure description language;

the list creating step of entering extracted logical structure information in a structure list; and

the definition creating step of creating a document type definition in accordance with the structure list.

8. The method according to claim 7, wherein said structure description language is SGML, and logical structure information is extracted from tags included in a sample document described in accordance with SGML in said structure information extracting step.

9. A recording medium wherein a program for executing each step in claim 7 by a computer is stored.

10. An apparatus for generating a document type definition of a structured document described in accordance with a structure description language, substantially as hereinbefore described with reference to any of Figures 1 to 14 of the accompanying 5 drawings.

11. A method of generating a document type definition of a structured document described in accordance with a structure description language as claimed in claim 7 and substantially as hereinbefore described.

10 12. A method of generating a document type definition of a structured document described in accordance with a structure description language, substantially as hereinbefore described with reference to Figures 1 to 14 of the accompanying drawings.

15 13. A recording medium as claimed in claim 7 and substantially as hereinbefore described.

14. A recording medium substantially as hereinbefore described with reference to any of Figures 1 to 14 of the accompanying drawings.



The Patent Office

3!

Application No: GB 9623606.2
Claims searched: 1-8,10-12

Examiner: B G Western
Date of search: 14 January 1997

Patents Act 1977 Search Report under Section 17

Databases searched:

UK Patent Office collections, including GB, EP, WO & US patent specifications, in:

UK Cl (Ed.O): G4A AUDB

Int Cl (Ed.6): G06F 17/21 17/27 17/30

Other: On-line databases: COMPUTER, INSPEC, WPI

Documents considered to be relevant:

Category	Identity of document and relevant passage			Relevant to claims
A	EP-0620527-A2	XEROX	See whole document	-
A	Dialog record 01805820 of Windows Sources, v3, n8, p128(3), August 1995, Frentzen J et al, "Company Policy".			-

X	Document indicating lack of novelty or inventive step	A	Document indicating technological background and/or state of the art.
Y	Document indicating lack of inventive step if combined with one or more other documents of same category.	P	Document published on or after the declared priority date but before the filing date of this invention.
&	Member of the same patent family	E	Patent document published on or after, but with priority date earlier than, the filing date of this application.